Working with Methods and Encapsulation - 0

Consider the code of Test.java file:

```
class Student {
     String name;
     int age;

     void Student() {
         Student("James", 25);
     }

     void Student(String name, int age) {
         this.name = name;
         this.age = age;
     }
}

public class Test {
     public static void main(String[] args) {
         Student s = new Student();
         System.out.println(s.name + ":" + s.age);
     }
}
```

What will be the result of compiling and executing Test class?

A - null:0

B - James:25

C - Compilation error

D - An exception is thrown at runtime

Working with Methods and Encapsulation - 1

What will be the result of compiling and executing Test class?

```
public class Test {
     public static void main(String[] args) {
         int x = 1;
         while(checkAndIncrement(x)) {
             System.out.println(x);
         }
     }

     private static boolean checkAndIncrement(int x) {
         if(x < 5) {
             x++;
             return true;
         } else {
             return false;
         }
     }
}
```

A - 2 3 4 5

B - 1 2 3 4

C - 1 2 3 4 5

D - Infinite loop

Working with Methods and Encapsulation - 2

What will be the result of compiling and executing Test class?

```
public class Test {
     public static void main(String[] args) {
         double price = 90000;
         String model;
         if(price > 100000) {
             model = "Tesla Model X";
         } else if(price <= 100000) {
             model = "Tesla Model S";
         }
           System.out.println(model);
     }
}
```

A - Tesla Model X

B - null

C - Compilation Error

D - Tesla Model S

Working with Methods and Encapsulation - 3

Consider the following class:

```
public class Employee {
     public int passportNo; //line 2
}
```

Which of the following is the correct way to make the variable ‘passportNo’ read only for any other class?

A - Make ‘passportNo’ static and provide a public static method getPassportNo() which will return its value.

B - Make ‘passportNo’ private and provide a public method getPassportNo() which will return its value.

C - Remove ‘public’ from the ‘passportNo’ declaration. i.e., change line 2 to int passportNo.

D - Make ‘passportNo’ private.

Working with Methods and Encapsulation - 4

What will be the result of compiling and executing Test class?

```
public class Test {
    public static void main(String[] args) {
        short [] args = new short[]{50, 50};
        args[0] = 5;
        args[1] = 10;
        System.out.println("[" + args[0] + ", " + args[1] + "]");
    }
}
```

A - [5, 10]

B - [50, 50]

C - An exception is thrown at runtime

D - Compilation error

Working with Methods and Encapsulation - 5

_____ uses access modifiers to protect variables and hide them within a class.

Which of the following options accurately fill in the blanks above?

A - Encapsulation

B - Abstraction

C - Polymorphism

D - Inheritance

Working with Methods and Encapsulation - 6

What will be the result of compiling and executing Test class?

```
class Message {
     String msg = "Happy New Year!";

     public void print() {
         System.out.println(msg);
     }
}

public class Test {
     public static void change(Message m) {
         m = new Message();
         m.msg = "Happy Holidays!";
     }

     public static void main(String[] args) {
         Message obj = new Message();
         obj.print();
         change(obj);
         obj.print();
     }
}
```

A -

```
null
Happy New Year!
```

B -

```
Happy New Year!
Happy Holidays!
```

C -

```
Happy Holidays!
Happy Holidays!
```

D -

```
Happy New Year!
Happy New Year!
```

Working with Methods and Encapsulation - 7

What will be the result of compiling and executing Test class?

```
public class Test {
     private static void m(int x) {
         System.out.println("int version");
     }

     private static void m(char x) {
         System.out.println("char version");
     }

     public static void main(String [] args) {
         int i = '5';
         m(i);
         m('5');
     }
}
```

A - int version int version

B - int version char version

C - Compilation error

D - char version char version

E - char version int version

Working with Methods and Encapsulation - 8

What will be the result of compiling and executing Test class?

```
//Test.java
class Student {
     String name;
     int marks;

     Student(String name, int marks) {
         this.name = name;
         this.marks = marks;
     }
}

public class Test {
     public static void main(String[] args) {
         Student student = new Student("James", 25);
         int marks = 25;
         review(student, marks);
         System.out.println(marks + "-" + student.marks);
     }

     private static void review(Student stud, int marks) {
         marks = marks + 10;
         stud.marks+=marks;
     }
}
```

A - 25-60

B - 35-60

C - 35-25

D - 25-25

Working with Methods and Encapsulation - 9

Consider code of Test.java file:

```
public class Test {
     public static void main(String [] args) {
         int [] arr = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
         String str = process(arr, 3, 8); //Line 5
         System.out.println(str);
     }

     /*INSERT*/
}
```

Line 5 is giving compilation error as process method is not found.

Which of the following method definitions, if used to replace /*INSERT*/, will resolve the compilation error?

A -

```
private static String process(int [] arr, int start, int end) {
  return null;
}
```

B -

```
private static int[] process(int [] arr, int start, int end) {
  return null;
}
```

C -

```
private static String[] process(int [] arr, int start, int end) {
  return null;
}
```

D -

```
private static int process(int [] arr, int start, int end) {
  return null;
}
```

Working with Methods and Encapsulation - 10

Choose the options that meets the following specification:

Create a well encapsulated class Clock with one instance variable model.

The value of model should be accessible and modifiable outside Clock.

A -

```
public class Clock {
  public String model;
  private String getModel() { return model; }
  private void setModel(String val) { model = val; }
}
```

B -

```
public class Clock {
  private String model;
  public String getModel() { return model; }
  public void setModel(String val) { model = val; }
}
```

C -

```
public class Clock {
  public String model;
}
```

D -

```
public class Clock {
  public String model;
  public String getModel() { return model; }
  public void setModel(String val) { model = val; }
}
```

Working with Methods and Encapsulation - 11

Following statement in a Java program compiles successfully:

student.report(course);

What can you say for sure?

A - student is the class name

B - report is the method name

C - student is the reference variable name

D - course must be of String type

Working with Methods and Encapsulation - 12

What will be the result of compiling and executing Test class?

```
public class Test {

     private static void add(double d1, double d2) {
         System.out.println("double version: " + (d1 + d2));
     }

     private static void add(Double d1, Double d2) {
         System.out.println("Double version: " + (d1 + d2));
     }

     public static void main(String[] args) {
         add(10.0, new Integer(10));
     }

}
```

A - An exception is thrown at runtime

B - Double version: 20.0

C - Compilation error

D - double version: 20.0

Working with Methods and Encapsulation - 13

What will be the result of compiling and executing Test class?

```
class Message {
     String msg = "Happy New Year!";

     public void print() {
         System.out.println(msg);
     }
}

public class Test {
     public static void change(Message m) {
         m.msg = "Happy Holidays!";
     }

     public static void main(String[] args) {
         Message obj = new Message();
         obj.print();
         change(obj);
         obj.print();
     }
}
```

A -

```
Happy Holidays!
Happy Holidays!
```

B -

```
null
null
```

C -

```
Happy New Year!
Happy Holidays!
```

D -

```
null
Happy Holidays!
```

Working with Methods and Encapsulation - 14

Consider the code of Test.java file:

```
class Student {
     String name;
     int age;

     Student() {
         Student("James", 25);
     }

     Student(String name, int age) {
         this.name = name;
         this.age = age;
     }
}

public class Test {
     public static void main(String[] args) {
         Student s = new Student();
         System.out.println(s.name + ":" + s.age);
     }
}
```

What will be the result of compiling and executing Test class?

A - Compilation error

B - James:25

C - null:0

D - An exception is thrown at runtime

Working with Methods and Encapsulation - 15

What will be the result of compiling and executing Test class?

```
//Test.java
class Point {
    static int x;
    int y, z;

    public String toString() {
        return "Point(" + x + ", " + y + ", " + z + ")";
    }
}

public class Test {
    public static void main(String[] args) {
        Point p1 = new Point();
        p1.x = 17;
        p1.y = 35;
        p1.z = -1;

        Point p2 = new Point();
        p2.x = 19;
        p2.y = 40;
        p2.z = 0;

        System.out.println(p1); //Line n1
        System.out.println(p2); //Line n2
    }
}
```

A - Point(19, 35, -1) Point(19, 40, 0)

B - Point(17, 35, -1) Point(17, 40, 0)

C - Compilation error

D - Point(17, 35, -1) Point(19, 40, 0)

E - Point(19, 35, -1) Point(19, 35, -1)

F - Point(19, 40, 0) Point(19, 40, 0)

G - Point(17, 35, -1) Point(17, 35, -1)

Working with Methods and Encapsulation - 16

What will be the result of compiling and executing Test class?

```
public class Test {
     public String name;
     public void Test() {
         name = "James";
     }

     public static void main(String [] args) {
         Test obj = new Test();
         System.out.println(obj.name);
     }
}
```

A - None of the above

B - null

C - Compilation error

D - James

Working with Methods and Encapsulation - 17

For the class Apple, which option, if used to replace /*INSERT*/, will print GREEN on to the console?

```
public class Apple {
     public String color;

     public Apple(String color) {
         /*INSERT*/
     }

     public static void main(String [] args) {
         Apple apple = new Apple("GREEN");
         System.out.println(apple.color);
     }
}
```

A - color = GREEN;

B - color = color;

C - this.color = GREEN;

D - this.color = color;

Working with Methods and Encapsulation - 18

What will be the result of compiling and executing following program?

```
class Rectangle {
     private int height;
     private int width;

     public Rectangle(int height, int width) {
         this.height = height;
         this.width = width;
     }

     public int getHeight() {
         return height;
     }

     public int getWidth() {
         return width;
     }
}

public class Test {
     public static void main(String[] args) {
         private int i = 100;
         private int j = 200;
         Rectangle rect = new Rectangle(i, j);
         System.out.println(rect.getHeight() + ", " +
        rect.getWidth());
     }
}
```

A - 200, 100

B - Compilation Error

C - 0, 0

D - 100, 200

Working with Methods and Encapsulation - 19

When does a class get the default constructor?

A - All classes in Java get a default constructor.

B - If the class does not define any constructors explicitly.

C - If you define parameterized constructor for the class.

D - You have to define at least one constructor to get the default constructor.

Working with Methods and Encapsulation - 20

Consider code of Test.java file:

```
public class Test {
     public static void main(String [] args) {
         int [] arr = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
          System.out.println(process(arr, 3, 8)); //Line 5
     }

     /*INSERT*/
}
```

Line 5 is giving compilation error as process method is not found.

Which of the following method definitions, if used to replace /*INSERT*/, will resolve the compilation error?

Select ALL that apply.

A -

```
private static String[] process(int [] arr, int start, int end) {
  return null;
}
```

B -

```
private static int process(int [] arr, int start, int end) {
  return null;
}
```

C -

```
private static String process(int [] arr, int start, int end) {
  return null;
}
```

D -

```
private static int[] process(int [] arr, int start, int end) {
  return null;
}
```

Working with Methods and Encapsulation - 21

What will be the result of compiling and executing Test class?

```
//Test.java
class Point {
    static int x;
    private int y;

    public String toString() {
        return "Point(" + x + ", " + y + ")";
    }
}

public class Test {
    public static void main(String[] args) {
        Point p1 = new Point();
        p1.x = 100;
        p1.y = 200;

        Point p2 = new Point();
        p2.x = 100;
        p2.y = 200;

        System.out.println(p1);
    }
}
```

A - Point(200, 200)

B - Point(200, 0)

C - Point(100, 200)

D - Point(100, 0)

E - Compilation error

F - Point(0, 200)

G - Point(100, 100)

Working with Methods and Encapsulation - 22

What will be the result of compiling and executing Greetings class?

```
public class Greetings {
    String msg = null;

    public Greetings() {
        this("Good Morning!");
    }

    public Greetings(String str) {
        msg = str;
    }

    public void display() {
        System.out.println(msg);
    }

    public static void main(String [] args) {
        Greetings g1 = new Greetings();
        Greetings g2 = new Greetings("Good Evening!");
        g1.display();
        g2.display();
    }
}
```

A -

```
Good Morning!
Good Evening!
```

B -

```
null
Good Evening!
```

C -

```
Good Morning!
null
```

D -

```
null
null
```

Working with Methods and Encapsulation - 23

Which of the following statement declares a constant field in Java?

A - int x = 10;

B - final static int x = 10;

C - static int x = 10;

D - const int x = 10;

Working with Methods and Encapsulation - 24

What will be the result of compiling and executing Test class?

```
public class Test {
     public static void change(int num) {
         num++;
         System.out.println(num);
     }

     public static void main(String[] args) {
         int i1 = 1;
         Test.change(i1);
         System.out.println(i1);
     }
}
```

A -

```
2
2
```

B - None of the other options

C -

```
2
1
```

D - Compilation Error

Working with Methods and Encapsulation - 25

What will be the result of compiling and executing Test class?

```
public class Test {

     private static void add(double d1, double d2) {
         System.out.println("double version: " + (d1 + d2));
     }

     private static void add(Double d1, Double d2) {
         System.out.println("Double version: " + (d1 + d2));
     }

     public static void main(String[] args) {
         add(10.0, new Double(10.0));
     }

}
```

A - Compilation error

B - Double version: 20.0

C - An exception is thrown at runtime

D - double version: 20.0

Working with Methods and Encapsulation - 26

What will be the result of compiling and executing Test class?

```
public class Test {
     static String msg; //Line 2
     public static void main(String[] args) {
         String msg; //Line 4
         if(args.length > 0) {
             msg = args[0]; //Line 6
         }
         System.out.println(msg); //Line 8
     }
}
```

A - Line 2 causes compilation failure

B - null

C - Line 8 causes compilation failure

D - Line 4 causes compilation failure

E - An exception is thrown at runtime by Line 6

Working with Methods and Encapsulation - 27

Which of the following can be used as a constructor for the class given below?

```
public class Planet {
}
```

A - None of the other options

B - public void Planet(int x){}

C - public void Planet(){}

D - public Planet(String str) {}

Working with Methods and Encapsulation - 28

Consider the code of Test.java file:

```
class Student {
     String name;
     int age;

     Student() {
         Student("James", 25);
     }

     Student(String name, int age) {
         this.name = name;
         this.age = age;
     }
}

public class Test {
     public static void main(String[] args) {
         Student s = new Student();
         System.out.println(s.name + ":" + s.age);
     }
}
```

There is a compilation error in the Student class.

Which modifications, done independently, print “James:25” on to the console?

Select ALL that apply.

A - Replace Student(“James”, 25); with super(“James”, 25);

B - Replace Student(“James”, 25); with this(“James”, 25);

C - Replace Student(“James”, 25); with this.Student(“James”, 25);

D - Add below code in the Student class:

```
void Student(String name, int age) {
  this.name = name;
  this.age = age;
}
```

Working with Methods and Encapsulation - 29

What will be the result of compiling and executing Greetings class?

```
public class Greetings {
    String msg = null;
    public Greetings() {
    }

    public Greetings(String str) {
        msg = str;
    }

    public void display() {
        System.out.println(msg);
    }

    public static void main(String [] args) {
        Greetings g1 = new Greetings();
        Greetings g2 = new Greetings("Good Evening!");
        g1.display();
        g1.display();
    }
}
```

A - Compilation error

B -

```
null
Good Evening!
```

C -

```
null
null
```

D -

```
Good Evening!
null
```

Working with Methods and Encapsulation - 30

Consider below code:

```
public class Test {

    private static void add(int i, int j) {
        System.out.println("int version");
    }

    private static void add(Integer i, Integer j) {
        System.out.println("Integer version");
    }

    public static void main(String[] args) {
        add(10, 20);
    }
}
```

Which modifications, done independently, print “Integer version” on to the console?

Select ALL that apply.

A - Replace add(10, 20); by add(10.0, 20.0);

B - Replace add(10, 20); by add(null, null);

C - Remove add(int i, int j) method declaration and definition.

D - Replace add(10, 20); by add(new Integer(10), new Integer(20));

Working with Methods and Encapsulation - 31

What will be the result of compiling and executing Test class?

```
public class Test {
     public static void print() {
         System.out.println("static method");
     }

     public static void main(String[] args) {
         Test obj = null;
         obj.print();
     }
}
```

A - static method

B - NullPointerException is thrown

C - Compilation error

D - None of the other options

Working with Methods and Encapsulation - 32

What will be the result of compiling and executing Wall class?

```
public class Wall {
     public static void main(String args[]) {
         double area = 5.7;
         String color;
         if (area < 7)
             color = "green";

         System.out.println(color);
     }
}
```

A - null

B - NullPointerException

C - Compilation error

D - green

Working with Methods and Encapsulation - 33

Consider below code:

```
public class Test {

    private static void add(int i, int j) {
        System.out.println("int version");
    }

    private static void add(Integer i, Integer j) {
        System.out.println("Integer version");
    }

    public static void main(String[] args) {
        add(10, 20);
    }
}
```

Which modifications, done independently, print “Integer version” on to the console?

Select ALL that apply.

A - Replace add(10, 20); by add(10.0, 20.0);

B - Replace add(10, 20); by add(null, null);

C - Remove add(int i, int j) method declaration and definition.

D - Replace add(10, 20); by add(new Integer(10), new Integer(20));

Working with Methods and Encapsulation - 34

What will be the result of compiling and executing Test class?

```
public class Test {
     public static void print() {
         System.out.println("static method");
     }

     public static void main(String[] args) {
         Test obj = null;
         obj.print();
     }
}
```

A - static method

B - NullPointerException is thrown

C - Compilation error

D - None of the other options

Working with Methods and Encapsulation - 35

What will be the result of compiling and executing Wall class?

```
public class Wall {
     public static void main(String args[]) {
         double area = 5.7;
         String color;
         if (area < 7)
             color = "green";

         System.out.println(color);
     }
}
```

A - null

B - NullPointerException

C - Compilation error

D - green

Working with Methods and Encapsulation - 36

Consider below code of TestBook.java file:

```
class Book {
    private String name;
    private String author;

    Book() {}

    Book(String name, String author) {
        name = name;
        author = author;
    }

    String getName() {
        return name;
    }

    String getAuthor() {
        return author;
    }
}

public class TestBook {
    public static void main(String[] args) {
        private Book book = new Book("Head First Java", "Kathy
         Sierra");
        System.out.println(book.getName());
        System.out.println(book.getAuthor());
    }
}
```

What will be the result of compiling and executing above code?

A -

```
null
null
```

B - Compilation error in TestBook class

C -

```
Head First Java
Kathy Sierra
```

D - Compilation error in Book class

Working with Methods and Encapsulation - 37

Consider below code of Circle.java file:

```
public class Circle {
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    public double getArea() {
        return Math.PI * radius * radius;
    }
}
```

User must be allowed to read and change the value of radius field. What needs to be done so that all the classes can read/change the value of radius field and Circle class is well encapsulated as well?

A - Change radius declaration from ’private double radius; to ’public double radius;

B - Nothing needs to be done

C - Add below 2 methods in Circle class:

```
protected double getRadius() {
  return radius;
}

protected void setRadius(double radius) {
  this.radius = radius;
}
```

D - Add below 2 methods in Circle class:

```
public double getRadius() {
  return radius;
}

public void setRadius(double radius) {
  this.radius = radius;
}
```

E - Change radius declaration from ’private double radius; to ’protected double radius;

F - Change radius declaration from ’private double radius; to “double radius;

Working with Methods and Encapsulation - 38

Consider the code of Test.java file:

```
class Report {
    public String generateReport() {
        return "CSV";
    }

    public Object generateReport() {
        return "XLSX";
    }
}

public class Test {
    public static void main(String[] args) {
        Report rep = new Report();
        String csv = rep.generateReport();
        Object xlsx = rep.generateReport();
        System.out.println(csv + ":" + (String)xlsx);
    }
}
```

What will be the result of compiling and executing Test class?

A - CSV:XLSX

B - XLSX:!XLSX

C - XLSX:CSV

D - An exception is thrown at runtime

E - Compilation error

F - CSV:CSV

Working with Methods and Encapsulation - 39

Consider below code of AvoidThreats.java file:

```
public class AvoidThreats {
    public static void evaluate(Threat t) { //Line n5
        t = new Threat(); //Line n6
        t.name = "PHISHING"; //Line n7
    }

    public static void main(String[] args) {
        Threat obj = new Threat(); //Line n1
        obj.print(); //Line n2
        evaluate(obj); //Line n3
        obj.print(); //Line n4
    }
}

class Threat {
    String name = "VIRUS";

    public void print() {
        System.out.println(name);
    }
}
```

What will be the result of compiling and executing AvoidThreats class?

A -

```
PHISHING
PHISHING
```

B -

```
null
VIRUS
```

C -

```
VIRUS
VIRUS
```

D -

```
null
null
```

E -

```
VIRUS
PHISHING
```

F - None of the other options

Working with Methods and Encapsulation - 40

Consider below code of Test.java file:

```java
public class Test {
    static int i1 = 10;
    int i2 = 20;

    int add() {
        return this.i1 + this.i2; //Line n1
    }

    public static void main(String[] args) {
        System.out.println(new Test().add()); //Line n2
    }
}
```

What will be the result of compiling and executing Test class?

A - It executes successfully and prints 20 on to the console

B - Line n2 causes compilation error

C - Line n1 causes compilation error

D - It executes successfully and prints 30 on to the console

E - It executes successfully and prints 10 on to the console

Working with Methods and Encapsulation - 41

Consider below code of TestSquare.java file:

```
class Square {
    int length;
    Square sq;

    Square(int length) {
        this.length = length;
    }

    void setInner(Square sq) {
        this.sq = sq;
    }

    int getLength() {
        return this.length;
    }
}

public class TestSquare {
    public static void main(String[] args) {
        Square sq1 = new Square(10); //Line n1
        Square sq2 = new Square(5); //Line n2
        sq1.setInner(sq2); //Line n3
        System.out.println(sq1.sq.length); //Line n4
    }
}
```

What will be the result of compiling and executing TestSquare class?

A - Compilation error

B - An exception is thrown at runtime

C - It prints null on to the console

D - It prints 0 on to the console

E - It prints 10 on to the console

F - It prints 5 on to the console

Working with Methods and Encapsulation - 42

Consider the code of Test.java file:

```
public class Test {
    private static void m(int i) {
        System.out.print(1);
    }

    private static void m(int i1, int i2) {
        System.out.print(2);
    }

    private static void m(char... args) {
        System.out.print(3);
    }

    public static void main(String... args) {
        m('A');
        m('A', 'B');
        m('A', 'B', 'C');
        m('A', 'B', 'C', 'D');
    }
}
```

What will be the result of compiling and executing Test class?

A - It compiles successfully and on execution prints 1333 on to the console

B - It compiles successfully and on execution prints 1233 on to the console

C - It compiles successfully and on execution prints 3333 on to the console

D - Above code causes compilation error

Working with Methods and Encapsulation - 43

Given code of Test.java file:

```
class Car {
    void speed(Byte val) { //Line n1
        System.out.println("DARK"); //Line n2
    } //Line n3

    void speed(byte... vals) {
        System.out.println("LIGHT");
    }
}

public class Test {
    public static void main(String[] args) {
        byte b = 10; //Line n4
        new Car().speed(b); //Line n5
    }
}
```

Which of the following needs to be done so that LIGHT is printed on to the console?

A - Delete Line n1, Line n2 and Line n3

B - Replace Line n4 with byte... b = 10;

C - Replace Line n5 with new Car().speed((byte...)b);

D - No changes are required as given code prints LIGHT on execution

Working with Methods and Encapsulation - 44

_____ modifier is most restrictive and _____ modifier is least restrictive.

Which of the following options (in below specified order) can be filled in above blank spaces?

A - public, private

B - default (with no access modifier specified), public

C - private, public

D - default (with no access modifier specified), protected

E - protected, public

Working with Methods and Encapsulation - 45

Consider below code of Test.java file:

```
public class Test {
    int i1 = 10;
    static int i2 = 20;

    private void change1(int val) {
        i1 = ++val; //Line n1
        i2 = val++; //Line n2
    }

    private static void change2(int val) {
        i1 = --val; //Line n3
        i2 = val--; //Line n4
    }

    public static void main(String[] args) {
        change1(5); //Line n5
        change2(5); //Line n6
        System.out.println(i1 + i2); //Line n7
    }
}
```

Which of the following statements are correct regarding above code?

Select ALL that apply.

A - Line n3 causes compilation error

B - Line n2 causes compilation error

C - Line n6 causes compilation error

D - Line n1 causes compilation error

E - Line n7 causes compilation error

F - Above code prints 8 on execution

G - Line n4 causes compilation error

H - Line n5 causes compilation error

Working with Methods and Encapsulation - 46

Given code of Test.java file:

```
public class Test {
    static String str = "KEEP IT "; //Line n1
    public static void main(String[] args) {
        String str = str + "SIMPLE"; //Line n2
        System.out.println(str);
    }
}
```

What will be the result of compiling and executing Test class?

A - SIMPLE

B - KEEP IT SIMPLE

C - KEEP IT

D - Compilation error

Working with Methods and Encapsulation - 47

Consider below code fragment:

private void emp() {}

And the statements:

1. Given code compiles successfully if it is used inside the class named ‘emp’
2. Given code compiles successfully if it is used inside the class named ‘Emp’
3. Given code compiles successfully if it is used inside the class named ‘employee’
4. Given code compiles successfully if it is used inside the class named ‘Employee’
5. Given code compiles successfully if it is used inside the class named ‘Student’
6. Given code compiles successfully if it is used inside the class named ‘*emp*’

How many statements are true?

A - Two statements

B - Three statements

C - Five statements

D - All six statements

E - Four statements

F - Only one statement

Working with Methods and Encapsulation - 48

Consider below code of TestMessage.java file:

```
class Message {
    String msg = "LET IT GO!";

    public void print() {
        System.out.println(msg);
    }
}

public class TestMessage {
    public static void change(Message m) { //Line n5
        m.msg = "NEVER LOOK BACK!"; //Line n6
    }

    public static void main(String[] args) {
        Message obj = new Message(); //Line n1
        obj.print(); //Line n2
        change(obj); //Line n3
        obj.print(); //Line n4
    }
}
```

What will be the result of compiling and executing TestMessage class?

A -

```
LET IT GO!
NEVER LOOK BACK!
```

B - Compilation error

C -

```
NEVER LOOK BACK!
NEVER LOOK BACK!
```

D -

```
LET IT GO!
LET IT GO!
```

E -

```
null
NEVER LOOK BACK!
```

F -

```
null
null
```

Working with Methods and Encapsulation - 49

Consider below code of Test.java file:

```
class Counter {
    static int ctr = 0;
    int count = 0;
}

public class Test {
    public static void main(String[] args) {
        Counter ctr1 = new Counter();
        Counter ctr2 = new Counter();
        Counter ctr3 = new Counter();

        for(int i = 1; i <= 5; i++ ) {
            ctr1.ctr++;
            ctr1.count++;
            ctr2.ctr++;
            ctr2.count++;
            ctr3.ctr++;
            ctr3.count++;
        }

        System.out.println(ctr3.ctr + ":" + ctr3.count);
    }
}
```

What will be the result of compiling and executing Test class?

A - 5:5 is printed on to the console

B - Compilation error

C - 15:5 is printed on to the console

D - 15:15 is printed on to the console

Working with Methods and Encapsulation - 50

Given code of Test.java file:

```
class Calculator {
    int calculate(int i1, int i2) {
        return i1 + i2;
    }

    double calculate(byte b1, byte b2) {
        return b1 % b2;
    }
}

public class Test {
    public static void main(String[] args) {
        byte b = 100;
        int i = 20;
        System.out.println(new Calculator().calculate(b, i));
    }
}
```

What will be the result of compiling and executing Test class?

A - 120.0

B - An exception is thrown at runtime

C - 5.0

D - 5

E - 120

F - Compilation error

Working with Methods and Encapsulation - 51

Consider below code snippet:

```
package com.training.oca;

public class Test {
    String testNo;
    String desc;
    /*
    Other codes...
    */
}
```

Which of the options are correct so that instance variables ‘testNo’ and ‘desc’ are accessible only within ‘com.training.oca’ package?

A - Change the instance variable declarations to: private String testNo; private String desc;

B - Change the instance variable declarations to: protected String testNo; protected String desc;

C - Change the instance variable declarations to: public String testNo; public String desc;

D - No changes are necessary